INTERACTIVE GRAPHIC SYSTEMS

GDS II

technical specification

CONTENTS

SECTION 1

INTRODUCING GDS II

In the intensely competitive semiconductor industry, rapid implementation of new technology means success. This demands a production design system that works swiftly and without interruption. CALMA's GDS II meets these criteria.

GDS II is a turn-key interactive graphics system that has been developed with the insight provided by years of experience with its predecessor, GDS. It is a turn-key system, yet it also offers unprecedented support to those users with the imagination to enhance its extensive capabilities.

Among the features and facilities described in the following pages are:

- 32-bit integer coordinate space to support VLSI
- Databases with information content extended beyond graphics
- User definable extensions to databases
- Design station featuring alphanumeric as well as graphic display
- Consistent, concise, and rich command language adaptable to both novice and expert operators
- GPL II™ Programming Language
- Comprehensive design rule checking programs
- Wide range of plotter support
- Exclusive All-Angle pattern generator support software
- State-of-the-art electron beam pattern generator support software
- Multiground, multitask, real-time disk operating system featuring:
  - Concurrent foreground and background activities
  - Concurrent production and program development
  - FORTRAN, ALGOL, DG/L, and assembly language user programming
- High performance ECLIPSE computer featuring:
  - Comprehensive instruction set
  - Extensive multiuser support hardware
  - Memory mapping
  - High speed floating-point processor
- New, unique erasable graphic display with continuous pan and zoom and instantaneous context switching

SECTION 2

GENERAL DESCRIPTION

2.1 The Problems of VLSI

GDS II is an advanced graphic system for the design of very large scale integrated (VLSI) circuits. Like its predecessor, GDS, GDS II provides a comprehensive turn-key solution to the problems of design, documentation, and artwork generation. But GDS II represents the next generation in interactivity, in efficiency, and in throughput.

VLSI presents several first order problems which cannot be solved by today's IC design systems. First, the required precision exceeds the capabilities of most existing systems. To resolve .1 micrometer features within a circuit 10 millimeters on a edge requires a resolution of 1 part in 10^5 - more than 16-bit integers can deliver. So GDS II uses 32-bit integer coordinates. Due to their uniform resolution, integer coordinate representations are, of course, superior to floating-point representations in applications requiring the generation of precision artwork.

Because of the complexity of VLSI, the sheer volume of data threatens to overwhelm a mini-computer based system. The GDS II Database Management System features unique mechanisms for data compression and data classification, minimizing disk requirements, both capacity and bandwidth.

The sheer volume of data poses a serious threat to background throughput. Background processes such as spacing checks and artwork generation involve sophisticated algorithmic processing and global sorting. To complete these tasks in an efficient and timely manner requires a powerful CPU and an efficient multiprogramming background facility which can make effective use of (relatively) large core configurations. The GDS II Background Processor provides significantly more throughput than its predecessor.

The sheer volume of data threatens to reduce the level of interactivity of the system. After all, it should take much longer to select and manipulate data within a complex VLSI circuit. WRONG! The GDS II display software, coupled with the VMD, CALMA's unique display terminal, provides the ultimate in comfortable and productive man-machine interaction.

2.2 The Solution - GDS II

Building on the latest in hardware technology and its long experience in the integrated circuit design market, CALMA has developed a comprehensive new system to solve the problems posed by the new integrated circuit technology. The GDS II database provides the ultimate in precision (32-bit integer representation). The Command Processor provides the highest level of interaction ever offered on a minicomputer-based system. GPL II™ is available, along with a comprehensive collection of background processes to drive Penplotters, Photoplotters, Optical Pattern Generators, E-Beam Pattern Generators, etc. Other background processes include extensive Design Rules Checks. The GDS II Background Processor provides for the efficient execution of multiple background tasks. For many tasks, GDS II will reduce the run time by large factors relative to GDS.

GDS II consists of a Central Processing Subsystem (including an ECLIPSE S/230 computer, an 80 Megabyte removable disk pack memory, a magnetic tape, and a system console) to which Design Stations, Plotters, and other peripheral equipment are attached. As many as six interactive, design stations may operate simultaneously, with minimal degradation in average system response time. At the same time, as many as three on-line plotters can be producing artwork at full rated speed, and GDS II can be writing a tape to drive an off-line plotter or pattern generator. These operations take place in full background mode, without interrupting input or editing.

To suit varying customer requirements, CALMA provides several input devices for GDS II work stations.

- CALMA 48-by-60 inch constrained-cursor digitizing table, with backlighting.
- Drafting-table-size tablet with puck and/or stylus.
- Console with 12-by-12 inch Rand-type tablet.

All stations include a keyboard, and an alphanumeric CRT (for designer communication). All are equipped with an interactive graphic display: An 11inch storage tube, a 19-inch storage tube, or CALMA's unique VMD.

Station configuration does not determine station function. Anything that can be done from one GDS II station can be done from all others.

And GDS II can read-in GDS databases. Thus existing designs created on GDS can be maintained on GDS II.

2.3 Exploiting The Latest Technology

Building on the experience gained over the past six years, during which GDS has become the standard of the semiconductor industry, GDS II incorporates the latest in hardware and software technology, beginning with the ECLIPSE S/230 computer.

The Eclipse Computer:

The ECLIPSE's high performance starts with a fast microprogrammed architecture. A random-access stack and versatile interrupt system replace software routines with high-speed hardware. A Floating Point Processor performs single and double-precision arithmetic at speeds matching those of large computers. Interleaving reduces memory cycle time. Effective memory speeds range from 200 to 640 nanoseconds.

These ECLIPSE computer features mean performance that translates into high throughput; throughput that lets GDS II handle the demanding tasks of interactive design, sophisticated rules checking, plotting, and user programming concurrently. Brielfy these features are:

- Comprehensive instruction set customized for operating systems and high-level language compilers. The set includes word, byte, and bit manipulation; extensive shift and logical operations; signed and unsigned integer multiply/divide; and block data movement.
- Extensive hardware stack mechanisms for fast subroutine linkage and context switching in real-time, reentrant environments.
- Memory Allocation and Protection with dynamic address translation, expansion to 512K bytes main memory, and write and address protection.
- High-speed Floating Point Processor featuring 32-bit and 64-bit formats and parallel processing.
- Memory interleaving and overlapping that significantly reduce instruction execution times.

CALMA's Multiground RDOS:

GDS II is built upon an enhanced version of Data General's Real Time Disk Operating System (RDOS). This more powerful operating system is known as Multiground RDOS because it supports additional jobs beyond the "foreground" and "background" provided by the standard Data General RDOS. Up to sixteen (16) separate "grounds" or jobs may be run concurrently with the system, providing complete isolation and protection for each ground. This permits, for example, several different interactive graphics systems to run at the same time (e.g., GDS II and DDM) or interactive graphics concurrent with program development and debugging.

RDOS permits "multitasking" within each ground; this means within the same address space there may be more than one asynchronous process (task) that competes for the CPU. The Scheduler for Multiground RDOS

is "interleaved" among grounds and tasks; that is a priority is computed for each task in each ground and then the system makes a global decision as to which task and ground with the priorities correctly interpreted.

Among the standard features of RDOS are Program Swapping, which allows up to five core images to be "pushed" to disk and them "popped" back and restored to execution. This permits multipass algorithms to be cleanly implemented in a manner that is transparent to the operator; also special subroutines such as sorting may be implemented as swaps so the entire core space of the ground is available for use.

The RDOS file system provides a generalized means of accessing disk files, at the same time supporting device-independent programming for characteroriented devices such as the DECwriter, Line Printer, and Station Keyboard. Filenames in RDOS may be from one to ten (10) characters in length and optionally may contain a one or two letter filename extension at the end. The filenames are referenced through directories which may be nested up to three deep.

Multiground RDOS retains complete compatibility with Data General's RDOS and therefore allows all the software products developed by Data General to be run without modification. Included are compilers for FORTRAN, ALGOL, BASIC, and DG/L. Text editors and a Macro Assembler are likewise available.

Multiground RDOS includes custom I/O drivers for all of CALMA's hardware. This permits a high level generalized interface for the graphics system eliminating the need to resort to bit level programming with critical timing conditions.

The VMD:

In addition to these second generation tools, CALMA has developed a new type of interactive graphics display which promises to provide a quantum jump in the level of user interactivity as well as a substantial improvement in overall system performance. Dubbed the Vector Memory Display (VMD) this device makes use of a vector memory and a digital TV imaging system. The result is a graphics display whose capabilities far surpass those of existing refresh displays, which use only one type of refresh memory (vector or raster).

Until now, CALMA has relied on a direct view storage tube display, which stores the graphics image directly on the face of a specially constructed cathode ray tube, so that the image does not have to be continuously refreshed. This approach results in a high-resolution, flicker-free image. In addition to the stored image, a graphics cursor is continuously displayed in write-through mode, so that it will not become a part of the stored image. Using a graphics tablet or digitizer as an input device, the user can point the cursor at objects on the screen and issue editing commands to the system to alter these objects or add new ones. On the whole this type of display has proved more than adequate for numerous applications in such diverse areas as integrated circuit and printed circuit design, cartography, and three-dimensional drafting, designing, and manufacturing.

However, the storage tube display does have one feature that limits its effectiveness in applications requiring a rapidly changing graphic presentation: the image cannot be selectively erased. Thus, viewing an altered graphics image requires erasing the entire old image and redrawing the entire new one. Although CALMA's storage tube display is many times faster than competitive displays of this type, applications exist for which the image cannot be repainted faster than the designer can think of what to do next.

To overcome this limitation requires a graphics display whose image is continuously being refreshed, so that when a graphic object is altered in the memory from which the display is refreshed, its image on the screen rapidly disappears and its altered image simultaneously appears, while the remainder of the image remains unchanged.

A display which refreshes the image directly from a "vector memory" meets this requirement. The display reads X-Y coordinate and intensity information from the memory and "strokes" the indicated line segments onto the screen in connect-the-dots fashion.

This approach, however, introduces more problems that it solves. For one thing, the complexity of the image which can be displayed without perceptible flickering is fundamentally limited - by how far the display tube's electron beam has to travel, how rapidly the beam can be deflected and modulated, and how rapidly the image disappears from the screen.

Various techniques could be used to sidestep these limitations, but not without creating further problems. For example, the length of the path traveled by the beam could be reduced by attempting to minimize the total length of the invisible segments, but the computational overhead would be substantial if not prohibitive. Likewise, the persistence of the tube's phosphor could be increased, but this would cause "ghost" images to remain on the tube for a longer time after they had been altered in the vector memory, thus returning to some extent to the basic limitation of the storage tube display. Moreover, as the deflection circuitry is pushed to its limits, difficulties in maintaining accuracy and repeatability are likely to manifest themselves, so straight lines appear other than straight and rectangles fail to close on themselves. Finally, since screen brightness and stroking speed are inversely related, any attempt to improve one parameter will compromise the other.

A display which refreshes the image from a "raster memory", or dot matrix, avoids the above problems. Flicker-free images can easily be generated, regardless of complexity, because the electron beam always travels the same path: a top-to-bottom sequence of closely spaced left-to-right lines, as in a commercial television set. The raster memory is used only to modulate the intensity of the beam.

Because the image-painting path is constant and relatively short, a bright image can be created; and because the path is so regular, the difficulties of maintaining accuracy and repeatability are vastly reduced. In addition, once the graphic vectors of the image have been "rasterized", i.e., converted to sequences of dots in the raster memory, the data is in the optimal order for displaying. The raster memory approach even makes it easy to create one image, erase a rectangular field within it, and then write another inset image in the vacated space. This can, of course, be done repeatedly if desired, creating a montage composite image impossible to achieve with a vector stroking display.

Unfortunately, displays which rely solely on raster memory for refreshing the image suffer from another basic limitation. Once a vector has been rasterized, there is no good way to deal with the resulting dots. If it is desired to remove only those dots corresponding to a given vector, one could rasterize the vector again and use the resulting set of dots to erase the raster memory selectively. But this would in general remove too many dots. What is really desired is to remove only those dots which the given vector was solely responsible for inserting and to leave alone those dots which were also inserted by intersecting vectors. But this is impossible, since in the raster memory all dots look alike. As a result, after the given vector is removed in this way, all conceivable intersecting vectors should be rewritten into the memory. In the worst case this amounts to rerasterizing the entire vector image, which runs the risk of nullifying the reason for going to a refresh display technology in the first place: the ability to alter the image rapidly.

The ability to zoom in on a selected portion of the data is best achieved by erasing the entire raster memory and rerasterizing the vector data at a different scale factor. This procedure brings out file detail that is unapparent at small magnification because different points must be mapped into the same dot. The alternative method of zooming, mapping each dot in a portion of the raster memory into a square array of dots on the screen, is a poor substitute, because the resulting magnification does not entail increased resolution.

What's more the raster-memory-only display offers no easy solution to the problem of selecting graphical objects on the screen by pointing at them. After all, a dot is just a dot. By contrast, a vector-memory-only display, for all its faults, finds this task relatively easy. A light pen detects it, causing it to be identified to the system.

CALMA's Vector Memory Display, by using both a vector memory and a raster memory, is able to realize the benefits and avoid the disadvantages of both. Its 21 inch, 1024-line video monitor displays a bright, flicker-free monitor (40 footlamberts - more than ten times brighter than typical storage tubes) of high resolution and accuracy, made possible by raster refresh technology. Yet, because of its vector memory, and the high-speed display processor which links it to the raster memory, the VMD can pan across, zoom in on, and zoom out from a graphical image in a continuous manner at frame rates as high as 40 per second. Furthermore, the instant selection of graphical entities is easily handled.

The VMD's vector memory utilizes a special purpose memory board, jointly developed by engineers at CALMA and Intel Corporation. This

board contains 65,536 16-bit words of solid state memory and can be accessed at a rate of one word every 300 nanoseconds. Each polygon entity stored within the memory requires two words for header information, between zero and three words for attribute information, two words for the initial X-Y coordinate pair, depending upon the representation mode. Thus the number of words used per vector could vary between one and nine, depending on the nature of the data; a realistic average is probably closer to two than three. Up to four memory boards can be configured in a VMD, for a total of 130,000 words, or roughly 50,000 vectors.

A command processor, which incorporates an 8085 microprocessor, handles all communication with the host central processing unit (CPU) and controls the operations of the various subfunctions of the pipelined display processor. In the input mode, polygon entities generated in the CPU are passed to the command processor through the memory update system to the vector memory. In the redraw mode, entities are fetched from vector memory, tested for overlap with the current display window, clipped to fit within the window (if need be), scaled, and imaged onto a portion of the raster memory. In the display mode, the screen is simply refreshed from the raster memory and the display processor is otherwise idle. In the identify mode, vector memory is searched at high speed and the header and attribute information pertaining to the polygon having a vertex closest to the indicated XY point is returned to the CPU. The VMD generates the images of up to four cursors and one display grid in hardware, so that they need not occupy space in the vector memory; similarly, three types of dashed lines can be generated in hardware, so that the individual visible segments need not be stored separately.

Under normal circumstances, in redraw mode, data are processed at the rate of one word (per vector memory board) every 300 nanoseconds, so that an entire drawing may be redrawn about 40 times a second, fast enough to give the impression of continuous movement on the screen. However, if the display window spans a large fraction of the data, the VMD redraw rate may be paced by the peak rasterizing rate, which is about 35 nanoseconds/dot along the major axis. Likewise, in the unlikely event that numerous diagonal vectors must be clipped to fit within the window, this process could momentarily pace the system. Sufficient data buffering is provided between system subfunctions to smooth out most local deviations from average data rates, thus assuring optimal system performance.

The raster memory used by the VMD features yet another unique memory board. It too was developed jointly by CALMA and Intel. This board contains an array of 512 x 512 bits of solid state memory. As the video monitor, which has a grid of 1024 x 1024 addressable points, is being refreshed, an interpolation algorithm, which operates in real time, is used to fill in the dots not explicitly stored in the raster memory.

In summary, the CALMA Vector Memory Display, by exploiting the best of two different refresh graphic memory systems, affords the designer an unprededented level of performance in refresh graphics displays. Features like unlimited inset capability, continuous pan and zoom, instantaneous context switching, and hardware select give GDS II an unequalled capability in its intended role as a VLSI design system.

The influence of the VMD on the overall design of GDS II is readily apparent. Menus are displayed on the screen, if desired, so that the designer never need load down at the tablet surface. All necessary command prompting is provided the instant it is needed and just as quickly erased when no longer required. Menus within menus can be used to explicate the valid choices available to the designer at any given time.

The designer can switch from viewing layer one of an integrated circuit mask set to viewing layer two - instantly, with no more than a token amount of assistance from the CPU. (By contrast, systems that use storage tube displays must rely on the CPU to read the entire disk area containing the drawing and extract the relevant information for redrawing. Clearly, the VMD approach leaves much more time for the CPU to service other graphics terminals and background tasks.) Likewise, one could view all lines of a given type, or all gates, or all contacts. It's all a question of masking in the desired attributes.

One of the most useful features enables the viewing of a large portion of an electronic circuit and simultaneously viewing in a graphic inset a highly magnified view of the same circuit, centered around the cursor position within the forementioned view. This arrangement proves

particularly useful in routing signal traces, where one needs to be aware both of the details of local obstacles and of the general direction in which one is headed. Modifications to the viewing parameters can be made independently from other commands and, in fact, within them. This enables the designer to route a trace to a different layer and then view the new layer of traces in solid lines and the old layer in dashed lines, all without terminating the input command.

MPC (not available in initial release):

In addressing the problem of checkplotting within the context of a VLSI design system, it became apparent that a high-resolution matrix plotter such as the Versatec 8242 offers a highly desirable solution. Integrating a device such as the 8242 into an interactive graphics system presents immense problems. CALMA has presently under development an intelligent Matrix Plotter Controller (MPC), built around the same technology as the Vector Memory Display, that addresses and eliminates these problems.

Consider the plight of a system architect attempting to integrate a matrix plotter into an ineractive graphic system. A Versatec 8242 electrostatic plotter is capable of producing a 40.96" x 40.96" plot at 200 spots/inch in slightly less than 82 seconds - regardless of complexity. Such a plot is 8192 x 8192 spots which requires 4,194,304 16-bit words to be represented in memory. This is exactly 128 times the capacity of a 32K word minicomputer which might seem the logical choice to perform the rasterization of data. Thus a 32K word minicomputer would require at least 128 passes of an unsorted drawing database to produce a plot image! Therefore, dedicating a minicomputer to the scan conversion task could at best be expected to produce a plot image in 128 times the view time. Thus, an IC that views in 10 seconds could take no less than 21 minutes to convert to a raster image. Other algorithms involving sorting, etc., introduce their own extensive resource requirements. A compute to plotting time ratio of over the order of 20 to 1 strongly suggests that the great speed potential of matrix plotters is an empty promise.

CALMA, however, seeking to realize the great performance and convenience benefits of matrix plotters for their customers, recognized that what these devices require is a special purpose controller to generate raster data at a rate to match their speed.

The ability to rapidly scan a plot file is the most critical requirement. The vector memory developed for CALMA's Vector Memory Display offers an attractive solution to this requirement. The MPC's vector memory utilizes a special purpose memory board, jointly developed by engineers at CALMA and Intel Corporation. This board contains 65,536 16-bit words of solid state memory and can be accessed at a rate of one word every 300 nanoseconds. Each polygon entity stored within the memory requires two words for header information, two words for the initial X-Y coordinate pair, and one or two words for each subsequent X-Y coordinate pair, depending upon the representation mode. Thus the number of words used per vector could vary between one and five depending on the nature of the data; a realistic average is between one and two. Up to four memory boards can be configured in an MPC, for a total of 262,144 words, or roughly 150,000 vectors.

Since the data in the vector memory can be processed in approximately 25 milliseconds, to drive the Versatec 8424 at 100% of rated speed requires that only 3 lines of raster data be generated per vector memory scan. To realize the goal of driving the device at rated speed, the MPC attaches high-speed special purpose computational hardware to the vector memory in a pipelined architecture.

Data leaving the vector memory is routed to a specially developed ALU that performs the rasterization of vectors. The ALU design features a microprogrammed approach, implemented in MECL, and utilizes 60-bit microinstructions with a cycle time of 75 nanoseconds. The ALU performs one arithmethic and one comparison operation during each cycle. The ALU also features multiply and divide hardware that operates independently to produce 16-bit integer results at a 40MHz rate.

The ALU executes a proprietary algorithm to perform the scan conversion and provide numerous improtant features. Line segments can be plotters at widths of 1 to 127 spots from center line data. Line segments may also be plotted as dashed, dotted or solid. Three patterns are also available to fill closed polygons. Fill patterns are contained in PROM's allowing them to be customized to user specifications. The area fill feature, of course, greatly enhances the utility of the generated plots by allowing areas to be readily distinguished.

Thus, the MPC makes the speed and attendent convenience of existing matrix plotters a reality to the users of interactive graphics systems. For the first time, a user can command a plot, wait a single view time to load the vector memory, and have his area-filled plot delivered at full rated matrix plotter speed.

SECTION 3

SYSTEM OPERATION

The operations of the GDS II system encompass a large variety of tasks in addition to simple digitizing. The shift in emphasis towards total design data bases, rather than simple graphics data bases, reflects the evolving requirements of the production environment as well as the design environment. The cabality to verify that final artwork conforms to design specifications, for example, requires that the information content of the data base be extended beyond strictly graphical data elements.

Since the GDS II System meets the users' requirements in doing a great deal more than merely collecting coordinates, the operation of the system must be discussed accordingly. The system provides a framework for the data collection process which each customer or installation may extend as desired. Before any digitizing may be done, the user must decide what data is to be collected, how libraries are to be organized, and so on. Only in this way can the most effective utilization of the system be achieved. Accordingly, the discussion of system operation begins with data base considerations.

3.1 Database Extensibility

CALMA provides turn-key systems in many areas of application. The GDS II system can be applied to a number of 2D disciplines including integrated circuit and printed circuit design. CALMA tailors the GDS II system to a particular discipline by implementing specialized primitves, implementing specialized GPL II programs, implementing specialized background programs, providing standardized menus, and providing standardized data bases. Each of these capabilities can be extended by CALMA or the customer. Extensibility facilitates both evolutionary system development and customization to meet application and user specifications. Of particular importance is data base extensibility, i.e., the ability to add new kinds of data base design with at most only externally defined (not understood by the system) conventions being used to implement the storage of new information. The GDS II Database Manager provides data base extensibility thus allowing the user to specify his own data base which incorporates elements and properties unique to him and meaningful to the system.

3.2 Database Elements

In dealing with large quantities of data two kinds of distinct partitioning are required. The first kind of partitioning is spatial. This kind of partioning allows extreme flexibility in specifying combinations of partitions for use in most editing operations. The second kind of partitioning allows the graphical presentation of an object to be specified independently of the spatial partitioning. Spatial partitioning is implemented by the 64 layers of a drawing. Representational partitioning is achieved through the 64 datatypes assignable to any element of the data base.

The GDS II data base organizes data into classes of elements:

(1) Paths - unclosed polygons with width

(2) Circuits - closed polygons

(3) Srefs - structure references

(4) Arrays - structure references in a rectangular configuration

(5) Text - describe the graphical presentation of paragraphs of text

(6) Snap - specifies connect points

(7) User-Definable - which allows the user to define the elements of the database, via convenient property lists

Graphic elements have layer and datatype associated with them.

The layer may be between 1 and 64.

The datatype may be between 1 and 64.

The path width may be any size.

3.3 Database Construction

A drawing in the GDS II System is defined as the contents of a library set. A library set is composed of a master library and a working library.

Master libraries are defined by the user on a project or process technology basis. The master library incorporates the data base schema and therefore determines the kind of data elements that may be defined in a working library. The master library may include exemplars which specify default values for typical working library elements. While an exemplar might be thought of as a skeleton or pattern for the data element, the data in an exemplar is accessed by reference. This allows the default values to be overridden on an instance-by-instance basis and yet permits the default values to be easily edited. Nested references by elements in the master library to structures defined with the master library are allowed.

A working library is always associated with a master library. Except for references to the master library, the working library is self contained. The working library may define exemplars and data structures for reference by name just like the master library. Any component unique to the working library is presumable defined in the working library. In particular, it is not necessary to define a component externally just to reference the structure at multiple locations in the drawing.

Given that a master library has been created, the typical use of the system is to create or edit a working library. While more than one master library may exist with multiple users accessing different master libraries, the master library may also be shared. While the master library would be handled as a read-only file in such a circumstance, a working library is generally allowed to be accessible by only one user at a time. In particular, multiple users cannot request update access to a single library.

3.4 Menu Operations

Menu Operations refer to the selection by the user of a string of characters to be processed just as if the user had typed them individually. Through this mechanism a unique "language" is implemented to interface operators or designers to the powerful primitives of the system. Because of GPL II programming these strings can be used to generate powerful, conditional, context dependent processing functions, complete with user prompts. At the other extreme each selection could generate just a single keystroke thus simulating a keyboard.

The implementation of the menu capability involves hardware and software selection mechanisms. Basically there are 512 different menu selections with as many as 64 characters definable to each. A 32 function button keyboard is provided, each key of which can select any of the 512 menu buttons. On a tablet or digitizing table a menu area is defined by a rectangular area divided into rows and columns of rectangles so that up to 512 of the menu buttons may be addressed by geometric position in the menu area. Additionally, the menu area may be shown on the VMD screen, so that when the cursor is in the area a selection may be made by moving the cursor to a user defined graphical presentation of a menu button and entering the selecting coordinate. By this method the user may keep his attention focused on the graphic display during system operation.

Creation and modification of menu buttons is easily achieved using the GDS II Text Editor (TED). Menu definitions may be saved and restored in RDOS files. Even though menus are easily produced, standard menus are provided to facilitate training.

3.5 Graphic Display

The Vector Memory Display (VMD) is an intelligent graphics display device which stores the display file in its own local memory. The display file is generated from the drawing data within the user-defined data base window of the drawing. All or part of the display file may be viewed on the

user-defined viewport(s) of the CRT screen. The viewport(s) may be located within the CRT window. The user can pan and zoom the data in the drawing file at any time, independent of digitizing or program execution.

The VMD has multiple cursors. Cursors may be used to trace the position of the digitizer (for coordinate entry) or the menu position for on-screen VMD buttons. Line types for paths/circuits include solid, dashed, dash-dotted, dotted, or invisible. The VMD has clean selective erase, which means that delete operations as they occur will be echoed properly. The user may place construction lines, which are temporary and do not exist in the data base, but which may be erased instantly at any time the user desires.

The VDM data base window is the actual contents of the vector memory of the VMD. A radical move over the big drawing necessitates reloading of the VMD. Typically, the view window will only be a portion of the database window.

Panning is either manual or automatic; the manual mode is invoked by moving the viewport in any direction or centering it on a spot. The size of the viewport is also user defined upon manual pan. In automatic mode, the system moves the viewport with the cursor, so that the cursor is visible at all times, for ease of digitizing without interruption.

Zooming in and out is done relative to the center of the viewport. Zoom may be direct or continuous; for direct zoom; the user specifies the absolute or relative magnification factor. The relative magnification factor is used for saying "2X" or "0.3X". In the continuous zoom, the user specifies the upper and lower absolute magnifications. To start the zoom before assigned completion, the user presses either button.

The user will normally have two viewports, one depicting the global view of the drawing, the other showing the local vicinity of the cursor while digitizing. The local may be in automatic pan and zoom modes, and the global view might be in manual pan and zoom. Moving the digitizer sensor (puck) moves the cursors in both views. If the user moves the cursor out of the local viewport, the automatic pan will continuously move the local view around the cursor.

For lesser display devices, the VMD emulator directs output to process GDS II manual viewing commands in as much as the device can act like the VMD. The emulator requires an extra RDOS ground and keeps the display file for the device on disk. The emulator cannot (in the case of the Tektronix CRT) partially erase, pan automatically, continuously zoom, or display multiple cursors.

Line Types include solid, dashed, dotted, dash-dotted, and invisible on the graphics display according to datatype specification. Four fill types are also defined and are displayable according to datatype specifications.

3.6 Character Display

The character display device supports the full ASCII character set, including lower case. The area of the screen is divided into two sections, a header in the first 3 lines and a scroll.

The header contains data which is updated continuously. The cursor location is displayed in user units on the top of the screen. This display is updated 10 times per second in parallel with the graphic cursor(s). The current time and date are also displayed in the header. The last line in the header is used for user-definable status information.

The remainder of the character display screen is used for a scroll with new lines being entered at the bottom of the screen. The scroll is being used to display input prompts, to log all commands executed, and to display any error messages.

In digitizing, if expected results do not appear on the graphic display, the user may redirect his attention to the alphanumeric display to find a record of the most recent interaction with the system. As a visual cue, all error messages will be output in bright characters.

A general capability featured in the GDS II system is the facility to "type-ahead". When a menu button is selected, for example, up to 64 characters including carriage returns are entered as a burst. All characters entered are echoed, but the echo is performed when the characters are actually used. It is conceivable that in some situations an advanced user will be

able to enter one or more complete commands while the system is still executing the current command. By delaying the echoing of any pending input, any output follows the command line which causes that output. Error messages are also synchronized with the command line which causes the error. If an error is detected, any input which is pending (and has not been echoed yet) is discarded.

All coordinates are equivalent to character strings. If the cursor is in the menu area, the display on the character CRT is the text associated with the button as described above. If the cursor is in the drawing area, however, the display on the character CRT is the current location of the cursor. The values displayed are in user defined units and represent the net effect of any scaling or working grid.

While the user will normally use the pen as the principal input device in a digitizing situation, he does have recourse to the inputting or editing of text data.

3.7 Coordinate Interpretation

It is important to separate the notion of physical units and the capability to resolve those units for representational and manipulative purposes. A given horizontal or vertical line can be measured in any units that the user deems natural. The number of addressable coordinates along the line are restricted to a finite number sufficient for all resolution deemed necessary by the user. A change of natural unit usually doesn't require a change in coordinate resolution. Sufficient resolution is mandatory. Most coordinates fall on a regularly spaced pattern called a working grid. Occasionally intermediate points are desired or necessary for specifying such things as half grid width for centers of lines with width or circle centers, more uniform representation of a circular boundary, or perhaps even a good representation of a square of some grid width rotated 45 degrees (n*sqrt(2) is irrational, but an additional 45 degree rotation really ought to be correct).

Because of such reasons as these and more importantly the need for higher density designs in smaller physical units, the GDS II system allows the user to specify any convenient natural unit and has a resolving power of one his natural unit resulting in an immediate change in the presentation of coordinates.

Because most coordinates fall on a regular grid patter, the system provides a working grid capability with independent X and Y spacing and origined at any coordinate of the data base. Each coordinate entered by the user is forced to the nearest grid point. Grids can also be displayed on the graphic CRT. Grids are readily changed on command.

For those using a scaled source document the axes of the document can be independently specified at any angle to the table with as many as 11 reference points on each axis used to scale the document.

3.8 Data Entry and Edit

The GDS II Graphics Editor contains three modes for the purpose of lightening the input task. The first is conventional straight-line edges. The second is "orthogonal interpolation", which outputs the segment as two pieces, the horizontal piece and the vertical piece. The user may specify X-first or Y-first. The final mode of input segment entry is "octagonal". The segments output are always horizontal, vertical, or 45-degree slant segments. Circles are additionally enterable as an interpretation mode by defining 3 points on the circumference, endpoints of a diameter, the center and the radius. Arcs are enterable as an interpretation mode in an even greater variety of ways.

Selection of items to change (one at a time) is done through the Graphic Editor via the get mode, which accepts one coordinate value.

Features possible for get mode:

1) get nearest sref
2) get nearest path
3) get nearest circuit
4) get nearest array
5) get nearest graphical element
6) get next identified element
7) get snap node definitions

One can choose the vertices to delete or add upon getting the polygon for edit. All operations done are abortable without the loss of information.

Elements are created in "modes" (such as path mode, circuit mode, sref mode, ...). This optimizes the amount of redundancy in user interaction. The modes are arranged in an English-language form for easy understandability, and ease of training.

Repetitious elements may be placed using copy.

Elements may be rotated, translated, magnified, or copied by the Graphic Editor.

Attributes of an element may be changed upon user command. (Such as layer, width, datatype, transformation...)

Polygons:

Polygons are created by specifying path or circuit as the current input mode, and entering its vertices in the same order that they occur in connection, and terminated by a null input line.

Polygons may be unioned, intersected, logically not-ed, and xor-ed with each other.

Any portion (or all) of a polygon may be repositioned with an arbitrary transformation, such as stretch, rotate, or mirror. Closure is preserved.

Rectangles:

Rectangles are optimized for entry, as two-point forms.

Structure References:

Structures are referenced by digitizing orgins in sref mode. User-definable transformations are saved as part of the structure reference.

Snap Elements:

Snap elements are defined and edited as a set of points. Snap locations are used to relate a structure to locations in its referencing environment.

Delete Features:

The user may delete points from the path/circuit in edit.

The user may delete srefs or arrays.

The user may delete the last thing he created: a general undo function.

The user may abort the current mode of entry.

The user may get a path/circuit and reopen it for edit, (then the user may undo vertices, or designate an arbitrary vertex or set of vertices for rapid deletion).

The GDS II Graphic Editor ignores trivial or redundant point cases to facilitate straightforward editing.

Exemplars:

The user may define any exemplar (property list) for part/circuit/sref/list/array binding.

Exemplar properties can be overriden by explicitly defined properties in a given element.

Properties in an element (layer, datatype, width, exemplar, reference, transform, etc.) can be changed by the user at any time.

3.9 Data Selection and Transformation

Editing multiple data base elements simultaneously involves selection criteria and transform operations. Most selections are on the basis of templets (selection masks), others are based on property selection, geometric selection, or identified order selection. The identified group is a set of elements accumulated through selection mechanisms. Once selected this group may be processed by the various editing operations

and still retain their identified property. Of course the identified group may be cleared at any time by the user.

"Templets" of layers are used to specify modifiable layers, addressable layers, and visible layers. Templets of data types are used to specify modifiable and addressable elements. Thus it is possible to restrict operations to any of 4096 (64*64) classes of elements or numerous other combinations. Templets are used to determine the four line types and area fill types for a given graphical presentation on the VMD display.

The system makes use of many selection mechanisms by their appropriateness and ease of use. Layer and data type templets are used most often in restricting various operations. Three other major methods of data selection are: property selection, geometric selection,and identified group selection. All of these methods are supported by each of the editing commands.

Property selection is determined by the specification of property relational statement composed of property specifiers and literal values combined with =, <, >,<=,>=,and,or,not. Such a statement, if true, selects the associated data base element.

Geometric selection is specified by the closest vertex to a point, all vertices/structure references inside/outside a given closed region, or all circuits/paths which are wholly inside/partially inside/partially outside/wholly outside a given closed area.

Identified group selection can be used to process all elements that are/are not in the identified group.

Edits do not modify the identified group except for deletions. The identified group may be cleared at any time. To add to/delete from the identified group any of the above methods of selection may be used.

Those selection methods addressing individual vertices may be used for stretching or deletion. More generally the addressing methods may be used to replace properties or to modify elements by translation, rotation, magnification, reflection, copying, deletion, changing layer, or changing data type.

3.10 Text Entry and Edit (TED)

The GDS II text editor is intended for many types of editing. The main types are:

1) Editing/Creating GPL II programs.

2) Editing/Assigning GDS II buttons for the menu.

3) Editing/Assigning text in a text element.

4) Editing/Creating Database Schemes.

Edit of a GPL II program is done by file name, edit of a button is done by button number, and edit of a text element is done by the location of the text element. Edit of a database schema is also done by name.

TED is a well-weathered Teco/style text editor, patterned after the one in RDOS.

Insertion/deletion of lines or characters is possible, as well as the powerful "change the next occurrence of this string to something else".

Searching for any string of characters, possibly containing a carriage return, is possible.

3.11 Initiating Background Jobs

The GDS II Job Management System provides the facility for executing background jobs. Any executable RDOS program ("save file") may be invoked as a background job. Background jobs are non-interactive: once started up, they execute independently of the initiating station.

Both CALMA-written and user-written programs may be executed as background jobs. The CALMA-written programs include the following: plotterdriving programs for on-line and off-line pen and photo plotters, pattern generator output programs for optical and electron-beam pattern generators, pattern generator read-in programs, dump and load programs for transferring drawings to and from magnetic tape, GDS-to-

GDS II data conversion, and design rule checking programs. These are described in detail in Section 5.5.

User-written programs can also be run as background jobs. These programs can be written in Algol or Fortran and compiled using Data General compilers. They may also be written in assembly language. Access to the GDS II database (for reading, creating or changing data) is available to Algol programs and Algolcompatible Fortran subroutines, through the use of CALMA-written database access routines.

The user initiates and controls background jobs through a number of GDS II commands. These commands allow the user to initiate a job, terminate or suspend a job, change the priority of a job, and find out information about all jobs in the system.

3.12 Program Development

The GDS II System supports a wide range of capability in the development of user programs. The development of custom programs is an option that allows the system to be tailored to the user's unique requirements in either the data collection or data processing areas. While no special programs are required to use GDS II System, even in many cases where the database definition has been extended, system enhancements may frequently be good investments.

The highest level programming language provided by the GDS II System is the GPL II™ Programming Language. The GPL II Language is a general purpose programming language designed for interactive use. It is patterned after APL, a programming language which is widely available commercially. A common characteristic of both of these languages is definition of a complete set of array operators. With the GPL II system, such operations as translation or rotation of a matrix representing a polygon is programmed as a single operation. Additional benefits include interactive execution and a full set of debugging aids.

The GPL II programming language is a superset of the GDS II command language. A set of commands can be transformed into a program with no changes whatsoever. Typically, the computational power of the GPL II Programming Language would be used to provide conditional execution of commands based on feedback from the system. On the other hand the computational power of the GPL II language can be used, for example, to evaluate arithmetic expressions in the command language without writing a program at all. A more significant use of the GPL II programming language is to implement programs which access values stored in user defined extensions to the data base.

The GDS II system supports several standard Data General programming languages. The most important of these is DG/L which is a derivative of ALGOL. Advanced users may choose to write DG/L programs that become part of the GDS II foreground system and may be referenced by the command language or the GPL II programming language. The user also has the option of writing a DG/L program which interfaces with backgroung job supervisor if less demanding access to the data base is sufficient.

The advantage in using DG/L is that the source language is compiled directly into the ECLIPSE instruction set rather than being interpreted as is the case for the GPL II programming language. Therefore, DG/L is a more suitable vehicle for programs with very extensive computational requirements. The disadvantage in using DG/L, as compared to GPL II, is that interactive debugging at the level of the source language is not possible. The physical limitations of the machine in respect to how much memory can be addressed at a time are also much more in evidence using DG/L.

The GDS II system also supports FORTRAN as defined by Data General. An interface between FORTRAN and the GDS II system can be implemented.

The GDS II System also supports the macro assembly language available from Data General.

Except for the GPL II Programming Language, all programming language development and debugging must be performed as a separate job. Provided that sufficient memory space is available, the user may detach his station from the GDS II System and enter the RDOS Command Line Interpreter. In this mode, all of the standard RDOS compilers, loaders, and utilities are available. Even debugging is possible except in those cases when the interrupt off debugger is required to diagnose a problem. At the conclusion of this mode of operation, the user

may reattach his station to the GDS II foreground and resume normal operations.

3.13 Command Primitives

Since the system is menu-driven, the actual command language format is relatively unimportant to either the casual or production user. At least during the get-aquainted period with the GDS II system, and perhaps for some time thereafter, the users' experience with the command language will be limited to the extent of defining new menu buttons to tailor the system to meet his special requirements. It is anticipated that some customers will see fit to create a programming staff to see to it that they are making the most efficient use of the system.

Since most of the interaction with the system is at the menu level, it has been possible in the design of the GDS II system to treat the command structure as a language. That is, a consistent set of syntactical rules have been defined that are simple to learn. The syntax of a command does not depend on the particular command in question. This approach has been possible only because we have not been forced to be overly concerned about keystroke optimization at the command level.

In addition to a simple and consistent syntax, the basic philosophy of the command language design is to separate the data collection, data manipulation, data storage, and command iteration functions. This design has advantages if the user does nothing but use the standard system menu.

The data collection routine for defining a closed polygon shape is a primitive function in the command language. The most important use of this primitive, of course, is to digitize a polygon to be saved in the data base. The same function may be used in defining a region for area editing or group selection commands, however. Traditionally, there have been several tools implemented for use as operator aids in digitizing complex shapes. As editing procedures have become more powerful, the user is left to define a complex window with no help at all. The GDS II polygon data collection primitive can be used to support both data creation and editing functions. Using the same primitive, of course, guarantees that the specification of a complex shape is always handled in a consistent method.

The data manipulation routines are implemented as another set of primitives. These routines are special in that a context is generally desired while digitizing. The variables in the context are important only to the data creation primitive. In the original GDS system, for example, changing the mask layer affects the operation of several commands, e.g. SS, RT, CC, etc. In the GDS II system, such a change affects only one primitive. The concept of mask layer is important only when the data is saved, not in how the data is digitized.

Finally, command iteration is separated from the actual function. Therefore, it is just as easy to define a menu button that iterates collecting circles as it is to define a button that iterates collecting rectangles.

3.14 GDS II Accounting

The accounting features included in GDS II facilitate the global supervision of users/projects, with consideration to how much foreground activity occurred for each user under whatever projects that user worked in. The time aquired will be listed in both wall-clock time (actual station time) and CPU time (computation time). Wall clock time reflects the actual time spent typing and waiting (if any) and CPU time reflects the time for GPL II programs run and GED primitives invoked. Provision is also made to log those background jobs which were started under the user/project, including the time for the run and a description of the command to generate the process, and, if applicable, when the job was aborted.

SECTION 4

SPECIFICATIONS

GDS II is a unique and complete graphics system featuring design and production capabilities. The following specifications exhibit its VLSI capabilities as well as more generally applicable features.

4.1 Database Features

Elements:

Elements are the basic entities with a GDS II Database.

Elements are assigned "element types" as they are created. The set of element types may be extended by the user.

Elements are collections of properties or attribute-value pairs. The data type, precision, transformation class, and shape of properties may be defined. The set of properties may be extended by the user.

Exemplars:

Exemplars are elements that have a user assigned name.

Once associated with an element, an exemplar supplies default or common properties thereby facilitating editing and also compressing data.

Structures:

Structures are collections of elements.

A structure is composed of a "superstructure" and "infrastructure". The superstructure contains only the data required to relate the structure to its external environment. The infrastructure contains the remaining information.

A structure may reference another structure in a hierarchial manner. Any reasonable number of levels of nesting structure references may be accomodated.

Libraries:

A library is a collection of structures, exemplars, schemas and tables for use by the Database Manager.

The number of structures in a library is limited only by available space.

Each library is implemented as an RDOS file and may contain up to 33.5 million bytes of storage area.

A working library together with its associated master library comprise a library set.

4.2 Input/Editing

Coordinates:

- Specified in user's natural units
- Natural and physical units changeable at any time
- Independently defined X and Y axes
- Independently scaled X and Y axes with up to 11 scaling points per axis

- Independently specified X and Y grid, scaled in user units, at any X and Y origin, changeable at any time
- Grid display on graphic CRT
- Straight, orthogonal, and octagonal interpolation
- Coordinates specifiable by digitizing, snapping, literal specification, and relative specification
- In elements, coordinates are connected by linear or circular interpolation
- Resolution of 1 part in 4,294,967,296

Selection:

- 64 layers and 64 datatypes specify up to 4096 classes of elements
- Templets of layers and other data types used to specify modifiable, addressable, and visible elements
- Selection by property of elements
- Selection by geometric relations
- Selection by membership in Identified Group
- Selection is used to add to Identified Group

Data Elements:

- Path - polygons with width
- Circuits - closed polygons
- Structure references
- Arrays of structure references
- Text
- Snap coordinates
- User-definable

Graphic Editor:

- Any element may be selected or created
- Properties of selected elements are modifiable
- Points can be moved, added, or deleted
- Selected elements may be rotated, translated, reflected, magnified, copied, or deleted

Data Transformation:

- All elements addressed may be rotated, translated, reflected, magnified, copied, or delted
- If geometric selection is made individual vertices may be moved or deleted
- Transforms affect elements, and, except for deletion do not affect membership in the Identified Group

Text Editing:

- Editing operations are the same for all text strings in the system
- Editing source may be a GPL II program, a menu button, any text property in an element, or a database schema definition

4.3 Display Control

Graphic Display - VMD:

The Vector Memory Display is used to display graphic data items, and, at the user's option, the current menu page. Data presentation features include:

- Line presentation may be solid, dashed, dash-dotted, or dotted
- Line presentation is a unique property--not necessarily related to layer
- Selected lines may be made invisible or blinked independently of presentation.
- Selected areas may be filled with pattern.

Alphanumeric CRT:

The alphanumeric CRT is used to display current status in real time and record a log of all recent operator interactions. Status information includes:

- Current cursor location, in user units, continuously displayed
- Current mode of interactive programs (if enabled)

Operator Interaction log includes:

- Commands executed
- Input prompts
- Error messages or results (if non-graphic)

Window:

The VMD is loaded with all data to be manipulated in an edit session. The result of this process, which may exceed 50K vectors, is termed the database window.

- Flexible data base window selector mechanisms:
  - Selection by property value--e.g. layer(s)
  - Selection by location
- Alternative handling of structure references:
  - Superstructure presentation--boundary and connection nodes
  - Infrastructure presentation--internal detail and nested reference

Viewports:

The VMD allows multiple, viewports into the data base window. Each viewport may be controlled independently of any other command in the system. Capabilities include:

- Increase magnification (and increase resolution)
- Decrease magnification
- Specify magnification factor
- Start/stop continuous zoom
- Move viewport in any direction
- Start/stop continuous pan

Update frequency:

- Viewport changes are handled instantaneously by the VMD
- All coordinates entered in the drawing area are marked immediatly, at the user's option on the graphic CRT

- All characters input as part of the current input request are echoed immediately
- All input typed ahead, prompts, error message, etc. are synchronized with the input commands
- Circuits are drawn as soon as each vertex is entered
- Paths are indicated by temporarily drawing the center line. The widened path is redrawn on completion
- Rectangles are drawn on completion
- Deletions are performed on command
- Cursor location(s) on the VMD and X-Y location displayed on the alphanumeric CRT are updated 10 times a second

Display Speed:

The GDS II Operating System is capable of drawing vectors while loading the VMD at a rate of 3,000 - 4,000 vectors per second. Any change to the display (pan, zoom, translate, modify) affecting a simple element or a group of elements is handled in 25 milliseconds. Assuming that the VMD memory is full, yields an effective transfer rate on the order of 100,000 to 200,000 vectors per second.

4.4 Background Programs

Design Rule Checks:

- Internal spacing checks
- External spacing checks
- Layer-to-layer intrusion, extension, oversize and area of intersecton (not available in initial release)
- AND, OR NOT combinations of layers (not available in initial release).
- Area Merging (not available in initial release)

Pen Plotters (on and off-line):

- Calcomp 936,960
- Xynetics 1100, 1200
- Versatec 8242 (not available in initial release)

Pattern Generators (input as well as output):

- Mann 1600, 2600, 3000, 3600
- Electromask 2000
- All-angle geometries may be processed

Photoplotters (not available in initial release):

- Precisionplot 2030
- Gerber S40, 632, 1232, 2032

E-Beam Pattern Generators (not available in initial release):

- Etec MEBES
- Cambridge EBMF

Database Compatability (input and output):

- GDS

4.5 Application Programming Tools

Graphics Programming Language:

GPL II™ provides:

- Numeric and character data types and operations.
- Scalar, vector, matrix, and higher order array data structures.
- Non-reverse list data structures.
- A full set of array and list processing primitives.
- An ALGOL-like control structure to facilitate structured programing.
- Full support of user programs--as operators or subroutines.
- Interactive execution--disk calculator mode.
- Interactive debugging at the source language level.

RDOS Programming Languages:

- FORTRAN
- ALGOL
- BASIC
- DG/L
- Macro Assembler

Utilities:

- Database Access Routines (FORTRAN, DG/L)
- Editors
- Loaders
- Debuggers

The combined facilities of GPL II™, FORTRAN, ALGOL, RDOS and the Database Access Routines offer the ability to tailor GDS II to a variety of graphic processing functions, including:

- Expansion of the command repertoire.
- Input/Output interface programming for special devices.
- Format conversion and manipulation of graphic data bases.
- Design rule checking.
- Simulation
- Test data generation

All programming and debugging is done on line, without jeopardizing circuit design or production activity.

4.6 HARDWARE

Central Processing System:	
Computer	ECLIPSE S/230
Instruction Length	16-bit and 32-bit
Hardware Accumulators	4
Index	2 hardware, 16 memory
Address Modes	Direct, Immediate, Indexed, Extended, and Multi-level Indirect Addressing.
Operator Control	Control panel with switch register
Memory Cycle Time	200-640 nanoseconds
Standard Memory Size	128K, 16-bit words (4 stations)
Memory Fully Expanded	256K, 16-bit words
Standard Arithmetic Element	High speed, single and double-precision floating point processor.
Standard Disk System:	
Transfer rate	2.1×10^6 bits/sec
Access time	35 ms (average random move)
Single drive capacity	12.5 million words
Expansibility	One additional drive may be added
Optional Large Disk Subsystem:	
Transfer rate	9.6×10^6 bits/sec
Access time	30 milliseconds (average)
Single drive capacity	50, 75, or 150 million words
Expansibility	Up to three additional drives of equivalent capacity may be added

4.6 HARDWARE (Continued)

Design Station:	
Tablet:	
Size	11" x 11"
Resolution	100 lines/in. or 0.01 in.
Accuracy	Λ0.01
Repeatability	Λ0.01
Input Device	Pen Stylus
Graphic Display (VMD):	
Vector Storage	128K 16-bit words of solid state memory store approximately 50K 16-bit vectors.
Imaging System	Digital TV refreshed 30 times per second, regardless of information on display.
Display Points	1024 x 1024
Display Size (diagonal)	21" (53.34 cm)
Point Generation	Any addressable point intensified.
Vector Generation	Hardware vector generation intensifies points between any two addressable points.
Image Storage	256,000 bits of solid state memory store value of each addressable point*
Storage time per image	Indefinite
Erase time	1 ms
Vector draw time	250 nanoseconds, plus 35 nanoseconds for each addressable point in major axis
Controller	Intel 8085 microprocessor

4.6 HARDWARE (Continued)

Alphanumeric Display:	
Full ASCII, 96-characters	A-Z, a-z, 0-9, puncuation and graphic symbols
Capacity	24 lines, 80 characters/line
Features	Addressable Cursor Reverse Video Bright Characters
Alphanumeric Keyboard:	
Full ASCII, 96-characters	A-Z, a-z, 0-9, puncuation and graphic symbols
Repetition Rate	10 cps

**As the video monitor, which had a grid of 1024 x 1024 addressable points, is being refreshed, an interpolation algorithm, which operates in real time, is used to fill in the dots not explicitly stored in the raster memory.*

SECTION 5

SOFTWARE

5.1 Multiground RDOS

The GDS II is built upon an enhanced version of Data General's Real Time Disk Operating System (RDOS). This more powerful operating system is known as Multiground RDOS because it supports additional jobs beyond the "foreground" and "background" provided by the standard Data General RDOS. Up to sixteen (16) separate "grounds" or jobs may be run concurrently with the system, providing complete isolation and protection for each ground. This permits for example, several different interactive graphics to run at the same time (e.g., GDS II and DDM), or a interactive graphics concurrent with program development and debugging.

CALMA's Multiground RDOS is arranged with a main-memory resident executive and system overlays residing on the disk. The executive must be resident in main memory storage before beginning continous and coordinated processing. The resident portion of RDOS can manage overlays and buffers, process system calls, and service device interrupts. The system overlay modules are dynamically brought into main memory from disk storage as required. These perform operations such as device/system initializations, file management functions like file creation/deletion and accessing, code conversions, real-time clock and time-of-day clock control and spooling control.

RDOS interacts with the user via the Command Line Interpreter (CLI). The CLI provides basic utility functions and file maintenance commands as well as a powerful means for passing parameters to programs that are invoked. Additionally, the CLI supports a nested macro facility that allows complex sequences of commonly used primitive commands to be parametrically generated in response to prompts from the user.

Memory Management and Protection:

A memory management and protection device has been developed for ECLIPSE computers. This option is supported under CALMA's Multiground RDOS.

The option extends the maximum main memory configuration for a single central processor from 32K to 256K words. Within the framework of an executing program, two modes exist. The first mode is the absolute mode. In this mode, only the lower 32K is directly addressable. RDOS resides in the low physical memory locations and executes in the absolute mode.

The second mode is called the mapped or user mode. In this mode, up to thirtytwo 1024_{10} word pages of logical memory are mapped into real memory. User programs execute in user mode. Thus, they need not be aware of their actual memory locations and need not occupy contiguous pages of memory.

I/O units can be selectively protected against unauthorized user control. Areas of logical memory can be write-protected and validity-protected on a page basis, allowing the supervisor to load a reentrant routine for many users only once. Validity protection insures that only that portion of the user's program being used need reside in storage.

Any program operating in user mode uses a complete logical address space, which includes its private page zero and extends through its upper memory bound. This is determined by the requirements of the individual program prior to its execution. Managing the mapping device and constructing the user program as a logical address space is also the responsibility of RDOS.

Among the standard features of RDOS are Program Swapping, which

allows up to five core images to be "pushed" to disk and then "popped" back and restored to execution. This permits multipass algorithms to be cleanly implemented as swaps so the entire core space of the ground is available for use.

RDOS also allows user direct I/O handling including data channel support and user interrupt handling. Multiground RDOS retains system protection features even when a user interrupt level routine is in control.

Task Scheduling:

A task is a logically complete execution path through a program. This path can be executed independently of another task within the same program.

RDOS permits "multitasking" within each ground; this means within the same address space there may be more than one asynchronous process (task) that competes for CPU. The Scheduler for Multiground RDOS is "interleaved" among grounds and tasks; that is a priority is computed for each task in each ground to give control to. This permits high and low priority tasks to co-exist in the same ground with the priorities correctly interpreted. The fast response of Multiground RDOS is attributed to this advanced scheduler that does not allow CPU-bound background operations to degrade the basic interactive response of foreground graphics. To fairly allocate the CPU among processes of equal priorities, the Multiground RDOS scheduler keeps statistics on which processes are using the most CPU time and correspondingly lowers their priority to allow equal progress by all processes with the same priority.

Input/Output Operations:

Efficient handling of input/output operations is an important feature of RDOS. A device-independent mechanism is provided to access all hardware file peripherals and disk program/data files.

Most data transfers to or from disk files and hardware devices are buffered by the operating system. Each system device handler has a small buffer associated with it, the size of the buffer depending on the speed of the device. Disk transfers are buffered in the system buffer area, which is organized into blocks of 256 words each. Data transferring from a disk file is read into this buffer area before the data within the block desired by the user is transferred to his buffer.

RDOS also provides a method for transferring data directly to or from the disk and buffer in the user memory area. This gives the user fast access to data stored on the disk.

Spooling:

When messages are output on a slow device like a teletypewriter, and its buffer fills up, the calling task is suspended until the buffer is emptied. Spooling allows messages to be temporarily stored on disk. The messages are later returned to main memory when space in the buffer is available. The significance of spooling is that output messages or information can be queued without putting excessive loads on the user main memory for buffering. This also frees the user from having to optimize his message requests, and thus permits more effective use of the device.

File System:

The RDOS file system provides a generalized means of accessing files on disk. Filenames in RDOS may be from one to ten (10) characters in length and optionally may contain a one or two letter filename extension at the end. The filenames are referenced through directories. Directories may be nested up to three deep in RDOS; the top level is equivalent to the entire disk pack and is called the Primary Partition. Files may exist at the primary partition level or may be associated with directories contained therein. The first level below the primary partition is called the secondary partition. It is characterized by a fixed amount of disk space. And it may either have files directly accessed from it or have one additional level of directory within called a subdirectory, which also may have files but no additional directory structures. Files in the current directory may be accessed by simply giving the filename; files in any other active directory (1-10 characters) followed by a colon to the filename.

RDOS provides three disk file structures: Sequential, Random and

Contiguous. Information in sequentially organized files is stored in groups of disk blocks. The last word of each 256 word block is used to store a link to the next block in the file. This link is invisible to the user, and is solely for the system use.

When building a sequential file, the system simply appropriates the next available disk block when storage is needed. It then constructs the link to the block. In a sequentially ordered file, after processing any given block, the system may step either to the previous block or to the next block in the series. Sequential files are very useful for indexes or small user files.

Random file organization provides the best combination of flexibility and accessibility of data. In randomly organized files, a master index of all physical block addresses is created. The master index blocks themselves are sequential files.

Blocks of data storage in random files utilize all 256 words for information storage. Each block is assigned a sequential positive integer by its position within the master index, indicating the block's logical position with the file. In processing randomly organized files, two disk accesses at most are required for the reading or writing of each block: one to access the file index and one for the block of data itself. If the index is main-memory resident (having previously been read into a system buffer), only one access is necessary.

Contiguous file organization has a rigid structure, but it provides the quickest access to data. Contiguous files are composed of a fixed number of disk blocks, which constitute an unbroken series of disk block addresses. These files can neither be expanded nor reduced in size, since, by definition, they occupy a fixed series of disk blocks. Contiguous files may be considered as files whose blocks may be accessed randomly, but without the need for a random file index.

All I/O operations which can be performed on randomly organized files can be performed on contiguously organized files. Contiguous files have the advantage of usually requiring less time for accessing blocks within the file.

Programming Facilities:

Multiground RDOS retains complete compatibility with Data General's RDOS and therefore allows all the software products developed by Data General to be run without modification. Included are compilers for FORTRAN, ALGOL, BASIC, and DG/L. To round out this collection of system programs Data General provides a complete selection of support programs such as editors, loaders and debuggers.

Programs may make calls to the GDS II Database Manager to access GDS databases.

Multiground RDOS includes custom I/O drivers for all of CALMA's hardware. This permits a high level generalized interface eliminating the need to resort to bit level programming with critical timing conditions.

RDOS Remote Communication Support:

CALMA RDOS currently supports remote communications through both asynchronous and synchronous data links. The asynchronous link operates at speeds up to 9,600 band and allows a remote station to function as an RDOS console through the QTY interface. Synchronous communications are supported at speeds up to 48K band. Currently software is available to provide both 2780 emulation and HASP Multileaving Work Station support for IBM 360/370 hosts or other machines using the same protocals.

5.2 Database Management System

Experience with GDS has demonstrated the need to shift from databases that concentrate strictly on graphics toward design databases. Objectives for the GDS II System, such as the capability to verify that final artwork conforms to design specifications, require that convenient means exist to expand the information content of databases beyond graphical data. Experience has also revealed the tendency for each application area (and even installation) to possess its own requirements that will evolve through time. Therefore, one of the primary design goals for the GDS II Database Management System is to provide the means for creation and maintenance of simple, flexible data structures that remain

extensible. Extensibility will facilitate both evolutionary system development and customization to meet application and user requirements.

To answer these needs, CALMA has developed the GDS II Database Management System to perform all database management functions - creation, modification, and retrieval - for the entire GDS II system. The DBMS is comprised of a set of procedures that perform these functions for both interactive and background processes. They are written in both DG/L and assembly language.

The type of information that may be placed in a given GDS II database is determined by a description file or "schema" that is referenced when the database is initialized. Each database that resides within a system may utilize a unique schema; thus, total flexibility is maintained. CALMA supplies a basic schema that can be augmented by the user to extend a database to contain the additional information required by a particular application.

Program maintainability is, of course, another critical area of concern in the design of the GDS II System. The design of the Database Management System greatly reduces and simplifies problems in this area by providing (only) "data independent" access to databases. That is, the DBM provides programs with access to logical data structures while internally managing all aspects of access to the corresponding physical data structures. Since programs will be written without dependence on physical data structures, they will remain immune to variations in those structures resulting from extended database definitions, etc.

Programs must, of course, always remain cognizant of the logical structure of the databases on which they operate. Special facilities are, however, included within the DBM that will allow programs to readily deal with extended databases.

Elements:

The basic components of GDS II database are its "elements". An instance of an element corresponds to an instance of a geometric entity such as a closed polygon or a reference to an entire collection of elements. Elements are partioned into classes as they are created by the assignment of an "element type" that has been previously declared in the database's schema. The assignment of element types allows programs interfacing with the database to immediately determine the appropriate manner to interpret a retrieved element. The DBMS may be generated to accept up to 255 element types.

Elements are defined in terms of their "properties".A property is an "attribute-value pair". An element, then is defined to be an arbitrary collection of properties. The primary means of database extension will be the user's ability to define new attribute-value functions to further describe the basic elements. For example, the DBMS provides the capability to add a "signal name= as a character string property to the elements representing interconnecting metalization. A program checking interconnection may then use the values of the signal name property in determining interconnection errors.

Property definition statements within a database's schema file determine the set of properties valid for that database. Figure 1 details the four characteristics of properties which are definable - data type, precision, transformation class, and shape. This figure also indicates which combinations are valid.

DATA TYPE	PRECISION (bytes)	TRANSFORMATION CLASS	SHAPE
INTEGER	1,2,4	ABSOLUTE RELATIVE COORDINATE	Scalar ARRAY (1 or 2 dimensional)
REAL	4, 8	ABSOLUTE RELATIVE	Scalar ARRAY (1 or 2 dimensional)
STRING	1 to 4095	Undefined	Scalar
AGGREGATE	(1 to 511)*2	Undefined	Single group ARRAY (1 dimensional)
ATTRIBUTES	1	Undefined	ARRAY (1 dimensional)
LIST	1 to 32K	Undefined	ARRAY (1 dimensional)

Note: A transformation class of COORDINATE is valid for ARRAY's of shape 2 x n and precision of 4 only.

Figure 1. Property Descriptions

As shown in the figure, six data types are currently defined:

1. INTEGER — a signed, two's complement integer of 8, 16, or 32 bits in length.

2. REAL - a single or double precision floating point number in standard ECLIPSE format.

3. STRING - a variable length string of characters (bytes).

4. AGGREGATE - a fixed number of 16-bit words.

5. ATTRIBUTES - a list of attribute indices.

6. LIST - a list of variable length byte strings.

Any definable property may be added to an element. Within an element scalar properties require 1 word (16 bits) overhead space plus their precision rounded to words for their storage. Arrays require 3 words of overhead per segment (segmentation of especially large arrays may be required due to page boundaries) plus the space required for their values at the stated precision. The DBMS processes and stores arrays as variable length lists of subarrays or single data items using only the amount of space required for actual data.

On retrieval coordinate data is automatically transformed by the DBMS according to a full linear transformation and translation specified by the calling program. Transformations are specified as a 64-bit floating-point angle of rotation in degrees, a reflection switch, and a 64-bit floating-point magnification factor. Each of these specifications may be designated as "relative" - i.e., to be composed when nested retieval occurs - or absolute. Translations are also specified as 64-bit floating-point X and Y values. The DBMS automatically composes and nests transformations as required.

Due to their high proportional population of the database, 32-bit coordinate data is also compressed by several procedures when stored within an element. 16-bit relative data is internally used when possible and may be further abbreviated to alternating X and Y increments if strictly axial vectors are being stored.

Thus the coordinates of an axial rectangle will occupy 8 words for data plus 3 words for overhead information. A widened line of n vertices will occupy from 4-(n#1) to 4n words plus overhead. The DBMS automatically handles all aspects of decompression and transformation of coordinate data that reside in the database.

INTEGER and REAL data representing linear dimensions are also automatically scaled by the system. Any property with a transformation class of "RELATIVE" is scaled by the maginification factor when retrieved.

A Database Manager may be generated to handle up to 255 property definitions. The standard system will accept 63 definitions.

Elements are stored as variable length record(s) within the pages (2048 byte blocks) of a library. Each element is assigned a unique "access key" when it is created. This key is a 22-bit value composed of a page number and a logical line number unique within that page. The DBMS provides the capability to access elements directly by their access key. Each such access requires a single page access within the database. The overhead associated with each element instance or element extension is 4 or 6 words.

The DBMS contains procedures that allow creation, modification, retrieval, and deletion of elements. As previously described, the Database Manager automatically scales, transforms, translates, compresses, decompresses, and adjusts the precision of values as directed by the database's schema and each respective calling program.

Exemplars:

Within a circuit design, elements normally fall with specific categories. All the elements representing interconnecting metalization are a good example. All these metal runs have properties in common yet unique to their own category. For this reason, the DBMS provides the means to allow elements to be factored in common and nonrecurring properties. This is accomplished through use of special elements called "exemplars". An exemplar is similar to an element in that it is also defined to be a collection of properties. It differs from a standard element in that each exemplar is assigned a user specified name at its time of creation. Exemplar names are from 1 to 32 alphanumeric characters.

The full definition of a standard element allows it to reference an exemplar by name. Thus, any standard element may be considered to consist of the union of its directly given properties and the properties of the referenced exemplar. Since all directly specified properties take precedence, the "standard specimen" which the exemplar represents may be flexibly modified on an instance-to-instance basis.

Exemplars are a great convenience to users; for once a standard specimen for a category is defined, all common properties are established. Editing the single exemplar will then effect the desired changes in all associated elements.

Because it is inherently a device for data compression, use of the exemplar has many positive side effects on the operational efficiency of the total system.

As in the case of elements, DBMS procedures exist that allow creation, modification, retrieval, and deletion of exemplars. A procedure also exists for renaming an exemplar.

Structures:

The DBMS also provides the means for grouping arbitrary collections of elements into "structures". A structure, then, is a collection of elements that is assigned a name at its time of creation. Structure names are from 1 to 32 alphanumeric characters. The DBMS maintains the time of creation and time of last modification for each structure.

Structures, of course, provide a convenient means for building hierarchies of geometries into increasingly complex assemblies. A structure is placed within another by creating an element that names the referenced structure and specifies its origin(s).

With this type of organization, it often becomes desirable to describe various properties of substructures parametrically. To provide this feature, the DBM allows a program to designate that the value of an attribute is to be determined by referencing a similar attribute at the next higher level of context, i.e., within the element that references the structure. The indirect reference may be made to a scalar property, an array element, or a list element.

The elements of a structure are partitioned into two sets: a "superstructure" and an "infrastructure". The elements comprising the superstructure should represent only that information which is useful for relating the structure to its external environment. For example, a dashed polygon outlining the boundary of a circuit along with the circuit's contacts would be a most useful representation of the component whenever it is desired to place it within a larger circuit. Thus, this abbreviated representation might naturally be chosen as the structure's superstructure. The infrastructure is composed of the remaining elements required for the full definition of the entity that the structure represents.

The Database Manager provides programs with access to either the superstructure, infrastructure, or both. Procedures exist for the creation, retrieval, deletion, and renaming of structures.

Libraries and Library Sets:

Collections of structures are organized by the DBMS into "libraries". The only number of structures that may be accomodated within a library is limited only by the available space. In addition to structures a library will contain a schema, tables for the use of the Database Manager, and possibly a collection of exemplars.

The structures within a given library may reference structures within a second designated library. The hierarchial organization of structures supported by the DBMS allows commonly used constructs to be placed within a "master library" and referenced by structures within any number of "working libraries". A working library together with its master library comprise a "library set". Libraries may be concurrently accessed by multiple processes in read-only mode. Thus, master libraries may normally be concurrently shared among interactive and background processes.

Each library is maintained as a single RDOS file with either random or contiguous organization. A single library may contain up to 16K pages (2048 bytes/page) or approximately 33.5 million bytes of storage area.

All file access is performed by procedures within the Virtual File Manager on a page basis. The Virtual File Manager greatly accelerates file access through sophisticated buffering techniques and also allows the sharing of a single copy of data among interactive processes.

5.3 GPL II™

The GPL II™ Subsystem includes a compilar for a high level application programming language, the required runtime support, and a growing number of CALMA developed application programs. While the language is based on APL, a commercially available programming language, the syntax has been redefined to serve as a natural extension to the GDS II Command Language.

Since a consistent syntax definition is used for both GDS II and GPL II™, and GDS II command is a legal statement in the GPL II™ Programming Language. Moreover, a GDS II command may make direct use of a GPL II™ function. For example, data collected by a GDS II system primitive may be passed to a GPL II™ function. And, data manipulated by a GPL II function may be passed to a GDS II system primitive.

The GPL II™ Language includes a large number of primitives designed for interactive use. The primitives are defined at a high level for efficient operator input and for efficient execution. One reason for writing GPL II™ programs is to further extend the number of primitives avialable for interactive use. While GPL II™ programs can generally be used in the same way that the built-in primitives are used, the programs can be tailored for more specific data processing requirements.

In addition to recieving data passed as parameters, GPL II™ programs and functions have direct access to global variables which define the station context. Most operating system peripherals (the notable exception is plotters) are fully supported for use by GPL II™ programs.

GPL II's computational capabilities are equivalent to those provided in APL/360 developed by IBM. Data may be organized as scalars, vectors, matrices, or higher dimensional arrays. Data may be stored in real, integer, logical, or character form. All storage allocation is performed dynamically by the GPL II™ Runtime Support. Automatic mode conversions are also supplied as necessary.

The GPL II™ Subsystem can be used at each station in a multi-station system. Separate work areas provide the capablility for each user to execute and debug programs independently of all other user. The interacive nature of the language simplifies programming and facilitates debugging. For example, one user has created a program that designs bipolar transistors given two parameters. The operator inputs the desired saturation resistance and emitter length. The program then informs the operator if any design rules have been violated. (A complete tutorial on how to use the program can be displayed on the CRT if desired.) The program calculates the collector size and creates the transistor in the database. The same program can also draw a graph on the CRT showing saturation resistance versus collector width. This program was written,

debugged, and documented for release within three days. And on a multistation system, concurrent production work could have proceeded normally on the other stations.

5.4 Background Job Management System

Background Processing Under GDS II:

CALMA currently has the largest library of background functions for integrated circuit applications in the interactive graphics industry. This extensive library includes design rule checks, plotter and pattern generator optimization programs and on-line plotter drivers. With the introduction of GDS II, these programs will be extended and new programs added.

The Job Management System supervises the execution of jobs, and efficient utilization of the on-line plotters. Background jobs are run in swappable grounds; this allows several background jobs to be simultaneously active using the same amount of main memory that a single job would require.

Features of the Background Job Management System include:

- INDIVIDUAL ADDRESS SPACES. Each backgound job has its own program address space. Because a job is physically unable to modify any memory locations other than its own, the integrity of other jobs and of the operating system is ensured.

- JOB QUEUING. Jobs are withheld from execution until the resources they need are available and until the current active job load is small enough to give them a reasonable expectation of access to the CPU.

- PRIORITY SCHEDULING. Jobs compete for CPU time on a priority basis, so that a high-priority job can be started at any time and not have to wait for other jobs that are already running.

- JOB BALANCING. The system scheduler dynamically accesses the extent to which each running job is CPU-bound orI/O bound, and takes this information into account to enhance job throughput.

- JOB SWAPPING. The Job Swapping Feature--unique to CALMA Multiground RDOS--allows the combined address space of all active jobs to be many times larger than the physical memory assigned to background activity.

- PLOTTER SPOOLING. The plotter spooler is specially adapted to plotter needs. The spool incorporates not only plotter data but also operator communications necessary to allow changing of paper and pens. This enables plotter spooling to be truly independent of job execution.

- DYNAMIC PRIORITY ADJUSTMENT FOR PLOTTER JOBS. The supervisor dynamically adjusts the priority of plotter jobs, so that jobs which have little data spooled are favored over jobs which have an ample amount of data spooled.

Nature of Job:

Any executable RDOS file ("save file") may be designated to be executed as a background job. Because they run in swappable grounds, background programs are prohibited from using certain system calls which pertain to real-time or interactive operation.

In particular, user-written programs in assembly language or in Data General compatible Fortran or Algol may be run as background jobs, provided they do not perform console I/O. Algol programs may access the GDS II data base using CALMA-provided data base access routines. (Data General DG/L permits the use of subroutines written in Fortran 5, subjuct to certain compatability conditions. This makes the data base access routines in effect available to Fortran programs also.)

User Interface:

The user communicates with the Job Management System through a set of job commands in the GDS II command language. The following commands are provided:

- ENTER JOB. This command allows the user to define a background job and enter it into the job system to be executed. The definition of the job includes the name of the save file to be executed and a string of parameters to be passed to the job. Alternatively, this command can invoke a foreground program which interactively collects and validates the parameters, prompting the operator for the required information. This interactive mode of job definition is used for plots, rules checks, and other jobs whose parameters are fairly complicated

- DISPLAY JOB INFORMATION. This command displays the status and related information for all jobs known to the system. The following information is displayed for each job:

  Job Name (both user and system name)
  Name of initiating user
  Status: waiting, active or complete; whether the job is suspended; whether it is an on-line plotter job
  Priority Class
  Time of Job Entry, Initiation and Termination
  Elapsed CPU Time for Job

- CHANGE PRIORITY CLASS. This command allows the operator to assign a different priority class to a job already entered.

- SUSPEND JOB. This command suspends the execution of an active job, or prevents a waiting job from becoming active.

- RESTART JOB. This command reverses the effect of the Suspend Job command and allows the job to be executed.

- KILL JOB. This command causes a job to be immediately terminated. An enqueued job is removed from the queue. All spooled output generated by the job is purged.

Job Identification:

Every job has two names by which it may be identified. One of these is the user defined job name, specified as part of the Enter Job command. Since the user defined name is not necessarily unique, the system assigns a unique system job name to each job at the time of entry. User job names are required to be longer than two characters, so as to distinguish them from the system job names, which consist of two characters only.

Re-Running Jobs:

Job parameters are stored in a job descriptor file, which persists for up to 24 hours after job termination. One of the options to the Enter Job command is to re-run a previously entered job. Thus, the job can be re-run without having to reenter all the parameters. Additionally, the user can create a non-temporary version of the job descriptor file, called a "job-save" file. This allows the job to be re-run at any future time.

Job Queuing:

When a job is entered, it is placed on a queue of waiting jobs. The execution of the job begins when the resources it needs are available. These resources include: magnetic tape units, on-line plotters, and also a program ground in which to execute. Grounds are assigned to jobs by the job scheduler in accord with the priority class of the job.

Job Priority Class:

When the user enters a job, he assigns it to a priority class. There are four priority classes, designated A,B,C, and D. Jobs should be assigned to classes according to their expected execution time, as follows:

Class	Expected Execution Time
A	Less than 3 minutes.
B	3 to 15 minutes.
C	15 to 30 minutes.
D	Longer than 30 minutes.

Each class has absolute priority over all lower classes. For example, if a class A job is entered while a class B job is executing, the class A job will pre-empt the class B job. The class B job regains control on completion of the class A job.

The above time limits are only suggested guidelines. Each installation may adopt its own policy regarding use of the priority classes.

On-Line Plotter Spooling:

Output to on-line plotters is spooled. That is, plotter data is generated in advance of the immediate needs of the plotter and is temporarily stored up in a disk file, called the plotter spool file, This enables the plotter to keep busy even if the plotter job is swapped out of main memory for an extended period of time. The spooled data is not limited to a single plot or plot job: even while a given plot is being produced, subsequent plot and messages to the operator are embedded in the spooled data as appropriate, to allow changing of pens and paper.

Scheduling of On-Line Plotter Jobs:

On-line plotter jobs have special characteristics in terms of job scheduling. Normally, all on-line plotter jobs are assigned to priority class B. This means that they tend to share time equally with each other and with any active class B job. However, their priority is subject to dynamic adjustment as a function of how full their spool files are. If a plotter's spool is empty or nearly empty, it will preempt other class B jobs. A plotter job whose spool becomes full is in effect suspended, thereby allowing other jobs to execute.

If a particular plotter job is lagging and must be expedited, it can be assigned to class A. It will then have absolute priority over all class B jobs, including other plotters.

5.5 BACKGROUND PROGRAMS

CALMA currently has the largest library of background functions for integrated circuit applications in the interactive graphics industry. This extensive library includes design rule checks, plotter and pattern generator optimization programs and on-line plotter drivers. With the introduction of GDS II, these programs are being extended and new programs added.

Pattern Generator Output:

The pattern generator output programs cause a magnetic tape for driving a pattern generator to be automatically generated, containing data from designated layers of a designated library drawing. Closed polygons, which may contain edges of arbitrary orientation, are efficiently fractured into rectangles. The entire set of rectangles for each layer is sorted and optimized for the particular pattern generator.

The following pattern generators are supported: Mann 1600, 2600, 3000 and 3600, Electromask 2000. All pattern generators are supported to their full resolution over the full range of stage travel.

The time to generate a tape (in the absence of other system activity) will not exceed the limits given below, for typical integrated circuit layers, whether of regular or random layout.

Number of Flashes	Time to Produce Tape
50,000	30 min.
100,000	1 hr.
500,000	5 hrs.
1,000,000	10 hrs.

For all these cases, the optimization will be such as to achieve an average pattern generator flash rate of at least 2 flashes per second for the Mann 3000, and at least 6 flashes per second for the Electromask 2000.

Pattern Generator Read-In:

The pattern generator read-in program reads any CALMA-generated pattern generator tape into the data base for verification purposes. The pattern generator models supported are the same as supported for output.

Photoplotter Output (not available in initial release):

GDS II photoplotter software supports photoplotters using aperture wheels, or a combination of aperture wheel and variable-aperture exposure systems. Closed polygons are automatically filled according to a variety of userspecifiable options. Filling of orthogonal figures can be done either wholly with circular apertures or with rectangular apertures to give sharper corners. Photoplotters connected directly to the system may be driven on-line; magnetic tapes are generated to drive off-line photoplotters.

The following photoplotters are supported: Precisionplot 2030, Gerber S40, 632, 1232, and 2032.

E-Beam Output (not available in initial release):

Programs will be provided to drive electron beam pattern generators, including the Etec MEBES, the Cambridge EBMF system, and others in response to user needs.

Pen Plotters:

The pen plotter programs drive on-line and off-line pen plotters. Currently supported on-line plotters are the Calcomp 960, 936, and Xynetics 1100 and 1200. Additional plotters will be added in response to user needs. Pen plotting features include:

- AUTOMATIC PEN SELECTION. At plot generation time, the operator can select which layers are to appear on a particular plot Normally, each layer is plotted with a different color. Single-pen plotters can be paused for a pen change between layers. Multiple-pen plotters can be made to automatically use separate pens for each layer, then pause when all pens have been used, or continue, reusing the same pens.

- SCALE. Plot scale is selected at plot generation time and is independent of data grid scale or input scale. This allows the user to plot at different scales, depending on the intended use of the plot.

- PLOT AREA. A plot can be of an entire drawing or an operator selected portion of the drawing. The area to be plotted is selected by defining a rectangular window.

- WIDENED LINES. Wide lines digitized in centerline form can be plotted showing either the center line or the outline of the intended trace or metal run.

- CROSSHATCHING. A crosshatching option permits automatic crosshatching of closed polygons.

GDS to GDS II Conversion:

This program reads a GDS-produced library or cell dump tape into the GDS II data base. (Note: Node type data will not be handled in the initial release.)

GDS II to GDS Conversion:

This program writes a designated library drawing onto a tape in GDS dump format. (Note: Because of the greater resolution and greater variety of data types found in the GDS II data base, an exact conversion of all data is not possible.)

Library Dump and Load:

The dump program dumps a library drawing to magnetic tape in standard GDS II dump format. The load program loads a drawing from such a tape.

Design Rules Checks:

GDS-EQUIVALENT RULES CHECKS.
The interior rules check tests closed polygon data for violations of specified minimum internal spacing. Optionally, the program can at the same time check for violations of a specified minimum notch width. All closed polygons on a specified layer within a specifid window are tested. In addition to a text summary of test parameters and results, errors are recorded in the form of a library drawing for visual inspection. This graphic error data may be displayed on the CRT or plotter either alone or in combination with original data.

The exterior rules check provides the same rules checking capability provided by the GDS Exterior Spacing Check. The check may be run within a single layer, or between two layers. The check finds all instances of spacing violations, which are defined as any pair of edges, belonging to distinct polygons, whose minimum straight-line distance from each other is less than a specified minimum. The violations are output in the form of a library drawing.

If a connectivity computation has been done on the layer(s), the option is available to report only violations between polygons which have distinct connectivity numbers.

ADVANCED RULES CHECKS (not available in initial release). The specification of the advanced rules checks will be developed in the first quarter of 1978. These checks will provide a variety of sophisticated checking capabilities, including: minimum layer-to-layer oversize, minimum layer-to-layer extension and intrusion, minimum layer-to-layer area of intersection.

Connectivity Computation (not available in initial release):

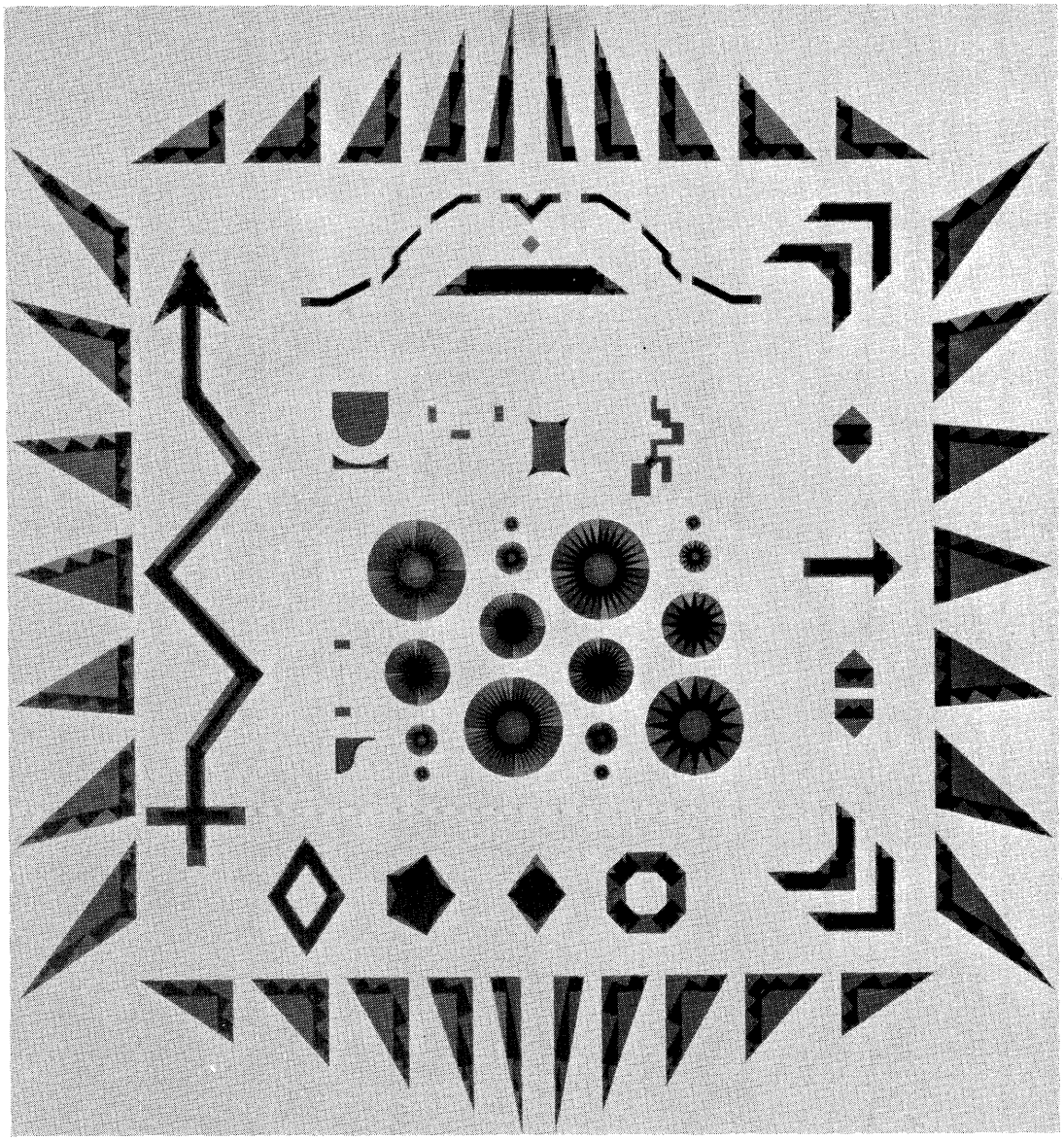
This program will determine the connectivity relations among all the polygons in a layer or set of layers, by assigning to each polygon an attribute, called "connectivity number", such that two polygons have the same connectivity number if and only if they belong to the same connected region.

Area Merging (not available in initial release):

This program will merge all figures comprising a connecting region into a single polygon or complex, over an entire layer or set of layers.

Logical Operations Between Layers (not available in initial release):

This program will compute the geometrical intersection (AND), union (OR), or difference of two designated layers. Both input and output layers must be in fully expanded form (no structure references).



These complex shapes, shown in this enlargement of a test pattern plotted on an Electromask Pattern Generator, demonstrate the highly sophisticated nature of CALMA's pattern generator software. The reticle was underdeveloped to illustrate how each shape has been fractured into a set of rectangles acceptable to the pattern generator.

SECTION 6

HARDWARE

6.1 Central Processing Subsystem

The GDS II has a Central Processing Subsystem to which Design Stations, Plotters, and other peripheral equipment are attached. The Central Processing Subsystem consists of a computer, a disk memory, a magnetic tape transport, and a system console. The Central Processing Subsystem can accomodate up to six Design Stations with as many as three Plotters.

Computer:

The GDS II computer is an ECLIPSE S/230 which accomodates up to 256K words of core memory. ECLIPSE computers have a microprogrammed architecture that incorporates a comprehensive instruction set. Instructions include:

- 16 bit and 32 bit instructions
- Direct, indirect, indexed, immediate and extended addressing
- Block, word, byte, and bit manipulation
- Fixed point signed and unsigned multiply/divide arithmetic
- Decimal arithmetic
- Extensive logical operations
- Single and double-word shifts, including hexadecimal shifts
- Single and double-precision floating point instructions
- Extended operation (XOP) for subroutines

Single instructions do work normally done by subroutines in operating systems and high-level language compilers. For example, the ECLIPSE computer DISPATCH ABSOLUTE instruction does most of the processing typically done by a FORTRAN computed GO TO statement. The LOCATE AND RESET LEAD BIT instruction lets ECLIPSE operating system quickly identify and allocate an available disk block. A BLOCK MOVE instruction moves data blocks between memory buffers. A complete set of logical operations and bit manipulation instructions is provided for data communications applications.

ECLIPSE computers use an extended last-in, first-out hardware stack facility to do operations complicated software algorithms normally do, but faster and more directly. An ECLIPSE stack is a series of variable-length temporary storage areas called frames, each easily assigned and randomly addressed. A single SAVE instruction allocates the frame, saves the entire machine state and sets a pointer to a frame area. A single RETURN instruction reverses the entire procedure. This ECLIPSE random access stack is far more efficient than traditional serial access stacks. It results in fast subroutine linkages, rapid context switching, and high system throughput in re-entrant high-level language and operating system environments.

The ECLIPSE computer's fast interrupt facility handles even the most timecritical events. Real-time operating system interrupt servicing is in hardware, rather than slower software routines. The servicing can be tailored to individual peripheral devices and applications with the powerful VECTOR instruction.

For simple interrupt processing, VECTOR combines fast service with minimum system overhead. The instruction identifies the interrupting device, then directly transfers control to the device handler, all in about 2.7 microseconds (4way interleaved core). For the most complete

interrupt processing, VECTOR identifies the interrupting device, switches from user stack to interrupt stack, saves the machine state, establishes a new priority, re-enables interrupts, then transfers control to the device handler. VECTOR typically performs even this most complex function in only 18 microseconds (4-way interleaved core).

The ECLIPSE Memory Allocation and Protection (MAP) provides program and data integrity in multi-user environments. It provides hardware protection between user programs, and between a program and the operating system.

MAP manages up to 512K bytes of main memory resource. It allocates memory to each user in 2K byte blocks, up to 32 blocks at a time. The blocks are small enough to make efficient memory use, yet large enough to keep system overhead to a minimum. The same block of physical memory can be allocated to more than one user, allowing procedure and data sharing among users.

MAP simultaneously holds three dynamic address translation maps - two user maps and a data channel map lets I/O activity of one user overlap with another's program execution.

In addition to address translation, MAP provides several kinds of protection. Blocks can be write-protected so users cannot alter them - an important feature when physical memory is shared. Input/output devices can be made accessible or inaccessible to a user. Each MAP protection function is enabled separately and easily, letting the operating system handle users with widely differing requirements.

The ECLIPSE Floating Point Processor performs extremely fast singleprecision and double-precision floating point arithmetic. High-level languages like Data General's FORTRAN 5 make extensive use of the Processor, producing high system throughput in computational applications.

The Processor has four separate 64-bit floating point accumulators for floating point arithmetic. Operands stored in these accumulators are quickly available for floating point manipulation, especially in interactive processes like sine and cosine calculations.

The Floating Point Processor operates in parallel with the computer's central processor. High-speed floating point arithmetic can therefore be performed simultaneously with other instruction processing.

There are 56 comprehensive floating point instructions that perform single - or double - precision floating point arithmetic.

Instructions PUSH or POP the entire Floating Point Processor state on or off the user stack, an extremely useful feature for rapid context switching.

FIX and FLOAT instructions allow convenient double-precision integer arithmetic with the Floating Point Processor.

The performance of ECLIPSE's Floating Point Processor is comparable to large computers. For example, a double-precision floating point ADD takes only 2.4 miroseconds, maximum with all but 1 microsecond overlappable with other CPU operations.

ECLIPSE computers interleave and overlap memories to increase speed and system throughput. Interleaving puts sequential memory locations on different modules. Core memories can be interleaved two, four, and eight ways. In four-way core interleaving, for example, each of four consecutive memory locations is on a separate memory module.

Since successive locations are on separate memory modules, and since ECLIPSE computer microinstructions are overlapped, ECLIPSE accesses the next sequential location before it completely finishes processing the previous location.

The effect on instruction execution time is dramatic. A four-way core interleaved system reduces a 1400 nanosecond LOAD ACCUMULATOR instruction to 800 nanoseconds. The more modules interleaved, the faster the memory cycle time. Users get higher performance when they need it most, in large memory configurations.

Disk Memory:

A removable-pack disk, with 12.5 million 16-bit words of memory for

database and program storage, provides an average access time of 35 milliseconds to the user's graphic database. The disk subsystem can be expanded to 25 million words through the addition of a second drive.

Optionally, CALMA can provide 80, 150, or 300 Mbyte Storage Modules. These removable pack disk drives should be used when extremely large files are to be created, or when there is a requirement for on-line storage of many files. Like the standard drive, the Storage Module is a moveable head, random-access disk drive. It uses a standard, eleven plotter, removable disk pack as the storage media. Average access time is 30 milliseconds. The Storage Module subsystem can be expanded to 4 drives of equivalent capacity.

Magnetic Tape Transport:

The magnetic tape transport is used for archival storage of databases and programs on 1/2-inch magnetic tape and to create tapes for off-line plotting. The standard drive is a 9-track unit, which is operated at 45 ips and 800 bpi. A 1600 bpi unit or combination 800/1600 bpi unit may be supplied in lieu of or in addition to the standard unit.

System Console:

The GDS II console is a keyboard printer with an ASCII keyboard (96 char). The printer is a 30 cps matrix printer with tractor drive and pin feed. It prints 132 characters per line, 10 characters per inch, and 6 lines per inch.

6.2 Design Stations

GDS II supports three types of Design Stations. The Constrained-Cursor Digitizer Station is generally used to interactively update existing designs or to interactively generate new designs. The Free-Cursor Digitizer Station offers the advantages of a tablet with the size of the Large Digitizer Table.

Constrained-Cursor Digitizer Station:

The Constrained-Cursor Digitizer Station comes equipped with a twin-stand, gantry-type digitizer with a seven-button cursor. The digitizer is back-lit with a variable intensity control. The resolution is 0.001 in.; the repeatability is ±0.001 in. The digitizing area is 46 x 60 in.

The station also includes an ASCII keyboard (96 char), XY-coordinate displays, an 11-inch storage tube display (for graphics) and an erasable alphanumeric display (for designer communications).

A 19-inch storage display may be specified in lieu of the smaller 11-inch display. And a 32-button programmable function keyboard may be added, if desired.

Tablet Station:

The Tablet Station comes equipped with a 12 x 12 inch rand-type tablet and stylus. The station also includes an ASCII keyboard (96 char), XY-coordinate displays, a 11-inch storage tube display (for graphics) and an erasable alphanumeric display (for designer communications).

A 19-inch storage display or CALMA's new VMD may be specified in lieu of the smaller 11-inch display. And a 32-button programmable function keyboard may be added, if desired.

VMD:

Vector Storage	128K 16-bit words of solid state memory stores approximately 50K 16-bit vectors
Imaging System	Digital TV refreshed 30 times per second, regardless of information on display
Display Points	1024 x 1024
Display Size (diagonal)	21" (53.34 cm)
Point Generation	Any addressable point intensified
Vector Generation	Hardware vector generation intensifies points between any two addressable points
Image Storage	256,000 bits of solid state memory store value of each addressable point*
Storage time per image	Indefinite
Erase time	1 ms
Vector draw time	250 nanoseconds, plus 35 nanoseconds for each addressable point in major axis
Controller	Intel 8085 microprocessor

**As the video monitor, which had a grid of 1024 x 1024 addressable points, is being refreshed, an interpolation algorithm, which operates in real time, is used to fill in the dots not explicitly stored in the raster memory.*

6.3 On-Line Plotters

On-line plotters supplied with GDS II include:

Calcomp Model 936 Drum Plotter:

33-inch drum; choice of 0.002 inch or .05 mm incremental step size; axial movement rate of 3.6 in./sec. in the pen down position and 5.0 in./sec. with the pen up; 3-pen plotting.

Calcomp Model 960 Plotter:

33 x 60 in. belt; .0125 mm incremental step size; axial movement rate of 30 in./sec. in the pen down position (10 in./sec. with liquid ink) and 30 in./sec. with the pen up; 2-pen plotting. The 4g acceleration permits the plotter to reach full drawing speed within a fraction of an inch. This capability improves plot times significantly.

Calcomp Model 748 Flat-Bed Plotter:

Flatbed plotter with 4-pen plotting at (axial) speeds up to 30 in./sec. 48 x 82-in. drawing area, ±0.005 in. absolute positioning accuracy over total drawing area, ±0.003 in. repeatability over total drawing area at 30 in./sec.

Xynetics Model 1050 Drafting Table:

Flatbed plotter with 4-pen plotting at speeds up to 35 ips, 33 x 45 in. drawing area, ±0.001 in./ft./axis absolute positioning accuracy over total drawing area, ±0.005 in. repeatability over total drawing area at 35 in./sec.

Xynetics Model 1100 Drafting Table:

Flatbed plotter with 4-pen plotting at speeds up to 35 ips, 42 x 57-in. drawing area, ±0.001 in./ft./axis absolute positioning accuracy over total drawing area, ±0.005 in. repeatability over total drawing area at 35 in./sec.

Xynetics Model 1200 Drafting Table:

Flatbed plotter with 4-pen plotting speeds up to 35 ips, 57 x 89 inch drawing area, ±0.001 in./ft./axis absolute positioning accuracy over total drawing area, ±0.005 in. repeatability over total drawing area at 35 in./sec.

Versatec Model 8242 Matrix Plotter:

Uses 42 inch roll paper. Dual array writing head has 8192 writing nibs. Plots with a resolution of 200 dots per inch at 0.50 inches per second paper speed.

6.4 Peripheral Options

GDS II provides a complete selection of peripheral options to meet varying user requirements. Included are a paper tape reader, a paper tape punch, a punched card reader, a line printer, and a communication interface.

Paper Tape Reader:

8 channels, 300 cps, supply and take-up reels, panel mounted.

Paper Tape Punch:

8 channels, 75 cps, supply reel, panel mounted.

Punched Card Reader:

80 columns, 285 cpm, table mountable.

Line Printer:

132 columns, 64 ASCII characters, 300 lpm, free standing unit.

Keyboard Printer:

Full ASCII (96 char) keyboard 30 cps, 132-column matrix printer, free-standing console.

Keyboard Display:

Full ASCII (96 char) keyboard, 30-480 cps, 2000 character video display, table mountable.

Synchronous Communication Interface:

To Bell 201 or equivalent. This synchronous interface enables Remote Job Entry emulation. With the appropriate software GDS II can emulate IBM's 2780 RJE Terminal, or IBM's HASP Terminal, or CDC's UT200 Terminal, or Univac's 1004 Terminal.